

**REMARKS**

The claims have been amended to clarify the scope of the present invention.

Claims 1-3, 7, 8, 12-17 and 19 have been cancelled and so will not be considered in the following remarks.

It is submitted that the claims as amended are clearly patentably distinguished from the cited references, whether considered individually or in combination.

Regarding the new **Claim 20**, it is believed that the present invention as claimed in this claim provides a novel and highly ingenious technique for handling intercommunication between processes in a simulation. In particular, the following concepts appear to be completely novel and original:

- Adding sender and receiver processes to a message data structure.
- Scheduling a message for processing when there is at least one sender process and at least one receiver process in its associated message data structure.
- Processing a scheduled message by calling the sender and receiver processes in its associated message data structure.

Rayner (EP 0 854 429) describes a simulation method in which a set of processes P1-P6 communicate by means of signal objects. A process object may change the state of one or more signal objects. A process object may in turn be dependent on (driven by) one or more signal objects (column 3, lines 31-35). An event queue 13 is used to schedule changes to the signal objects.

However, there is no suggestion in Rayner of associating a message data structure with a message, to which sender and receiver processes are added as recited in claim 20 steps (b) to (d). In Rayner, when a process requires to send a signal, it updates the signal object and then calls the `send_event()` function of the signal object, which in turn schedules the change on the event queue. Thus, Rayner clearly does not add sender and receiver processes to any data structure associated with the signal.

Furthermore, there is no suggestion in Rayner of scheduling a message when it has at least one sender process and at least one receiver process in its associated message data structure, as recited in claim 20 step (e). On the contrary, in Rayner, when a process requires to update a signal object, the change is scheduled in the event queue, without waiting for any receiver process.

Furthermore, there is no suggestion in Rayner of processing a scheduled message by calling its associated sender and receiver processes, as recited in claim 20 step (f). On the contrary, in Rayner, a change scheduled in the event queue is processed by `pop_events()` which calls the `fire_event()` function of the signal object, which in turn calls the `entry()` function of the dependent process. In other words, processing of the scheduled change indirectly calls the dependent process, but clearly does not call the sender process.

The examiner suggests that Lutter ("Using VHDL for Simulation of SDL Specifications" IEEE 1992) teaches providing a message data structure for messages, and that when a process requires to send a message, the process is assigned to the relevant message data structure as a sender, and when a process requires to receive a message, the process is assigned to the relevant message data structure as a receiver. However, the applicant respectfully disagrees. It is pointed out that Lutter teaches that every process has an input queue in which the arriving signals are stored (page 630, column 2, lines 29-31). Therefore, rather than adding processes to message data structures, Lutter adds signals to process data structures, which is a completely different thing.

Hence, it is respectfully submitted that new claim 20 is clearly patentable over the combination of Rayner and Lutter.

New **Claim 21** is similar to claim 20, and it is submitted that it is patentable for the same reasons.

**Claim 4** recites that each message data structure includes a sender queue and a receiver queue for queuing a plurality of sender and receiver processes.

The examiner suggests that Lutter teaches a sender queue and a receiver queue for queuing a number of sender or receiver processes. However, it is respectfully submitted that the only queues described by Lutter are the input queues in which the arriving signals are stored for processes. Thus, Lutter queues signals, not processes, and hence clearly does not queue sender and receiver processes.

Hence, it is respectfully submitted that claim 4 is clearly patentable over the combination of Rayner and Lutter.

**Claim 5** includes checking the message data structure when a message has been processed, to determine whether there is at least one remaining sender and receiver in the message data structure for the message and, if so, rescheduling the message.

The examiner suggests that Lutter teaches that sending signals to other processes encompasses rescheduling the message if a remaining sender and a remaining receiver for the message exist. However, the applicant respectfully disagrees. As shown above, Lutter does not describe a message data structure containing queues for sender and receiver processes: the only queue described by Lutter is for signals, not processes. Hence, there is no message data structure to check to determine whether there is at least one remaining sender and receiver for the message.

Hence, it is respectfully submitted that claim 4 is clearly patentable over the combination of Rayner and Lutter.

**Claims 6 and 9** were rejected as unpatentable over Rayner and Lutter in view of Hashmi (U.S. Patent 6,161,081).

It is agreed that Hashmi teaches composition and decomposition of messages. However, it is respectfully submitted that Hashmi clearly does not teach a message data structure holding pointers to a composition activity and a decomposition activity.

It is also respectfully submitted that Hashmi clearly does not teach that the decomposition activity is activated when a process is added as a sender for a message and the composition activity is activated when a process is added as a receiver. Hashmi teaches that

"When unit U1 requires to send a message to units U2 and U3, it sends a series of bytes to the arbiter 56 of interface 11. Each byte is accompanied by a Decompose request" (column 5, lines 19-20).

Thus, in Hashmi, decomposition is activated when a byte is received by the interface 11, not when a process is added to a message data structure.

Therefore it is respectfully submitted that even if the teachings of Rayner, Lutter and Hashmi were combined, the combination would not have suggested the invention as claimed in claims 6 and 9 to a person skilled in the art.

**Claims 10 and 11** were rejected as anticipated by Rayner. However, it is respectfully submitted that these claims are clearly distinguished from the teachings of Rayner.

The examiner suggests that Rayner teaches scheduling processes and messages by placing process type items and message type items on the scheduler queue. However, the applicant respectfully disagrees. In Rayner, it is clear that the event queue and the delta queue are used only to schedule changes of state of signal objects (see column 5, lines 29-31 and column 6 lines 3-5).

Also, regarding claim 11, it is respectfully submitted that Rayner does not teach scheduling messages by calling the sender and receiver processes. As shown above, Rayner teaches that processing of a scheduled change calls the dependent process, but clearly does not call the sender process. Moreover, it is respectfully submitted that Rayner does not teach scheduling processes and messages by placing process-type items and message-type items on the scheduler queue. As shown above, Rayner schedules only changes to signals, and does not schedule processes.

**Claim 18** was rejected as unpatentable over Rayner in view of Wilkes ("Application of High Level Interface-based Design to Telecommunications System Hardware", ACM 1999).

It is respectfully submitted that Rayner does not teach scheduling messages by calling the sender and receiver processes. As shown above, Rayner teaches that processing of a scheduled change calls the dependent process, but clearly does not call the sender process. Moreover, it is respectfully submitted that Rayner does not teach scheduling processes and messages by placing

process-type items and message-type items on the scheduler queue. As shown above, Rayner schedules only changes to signals, and does not schedule processes.

The examiner also suggests that Wilkes teaches items on a scheduler queue with a type value `q_cmd_t`. However, it is respectfully pointed out that what Wilkes actually says is that "the message called `cmd_m` has one parameter of type `q_cmd_r`". However, there is clearly no suggestion in this that `q_cmd_t` indicates the type of an item in a queue, and in particular whether that item is a process-type item or a message-type item.

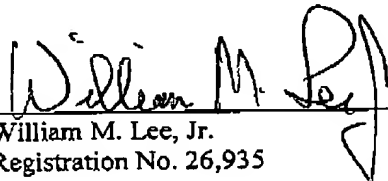
Therefore it is respectfully submitted that even if the teachings of Rayner and Wilkes were combined, the combination would not have suggested the invention as claimed in the amended claim 18 to a person skilled in the art.

#### *Conclusion*

In summary, it is submitted that this application is now clearly in order for allowance and such action is respectfully solicited.

Respectfully submitted,

Date: August 9, 2004

  
\_\_\_\_\_  
William M. Lee, Jr.  
Registration No. 26,935

Barnes & Thornburg LLP  
PO Box 2786  
Chicago IL 60690-2786  
(312) 357-1313 Telephone  
(312) 759-5646 Facsimile  
(312) 214-4811 Direct